PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:		
Mark D. Rustad		Examiner: Justin King
Serial No.:	09/437,169	Group Art Unit: 2181
Filed:	November 10, 1999	Docket: 977.029US1
For:	SYSTEMS, DEVICES, STRUCTURES, AND METHODS TO SHARE RESOURCES AMONG ENTITIES	

APPELLANT'S BRIEF ON APPEAL

MAIL STOP APPEAL BRIEF – PATENTS
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313 1450

Sir:

This brief is presented in support of the Notice of Appeal filed on January 20, 2004, from the final rejection of pending claims 1-5, 8, 11-20, 23-31, 34-38, 41-46, 49, 52, 55-64 and 67-77 of the above-identified patent application. The Office Action from which Appellant appeals was mailed September 18, 2003.

The Appeal Brief is filed in triplicate. Please charge the brief filing fee of $165.00 to Deposit Account No. 19-0743.

Appellant respectfully requests reversal of the Examiner's rejection of pending Claims 1-5, 8, 11-20, 23-31, 34-38, 41-46, 49, 52, 55-64 and 67-77.

APPELLANT'S BRIEF ON APPEAL

TABLE OF CONTENTS

Real Party in Interest

The real party in interest of the above-captioned patent application is the assignee, Digi International, Inc.

Related Appeals and Interferences

Appellant knows of no other appeals or interferences which will have a bearing on the Board's decision in the present appeal.

Status of the Claims

Claims 1-5, 8, 11-20, 23-31, 34-38, 41-46, 49, 52, 55-64 and 67-77 are pending; all of these claims have been finally rejected, and are the subject of the present appeal.

Status of the Amendments

The original claims are 1-79.

On October 10, 2002, Appellant cancelled claims 6, 7, 9, 10, 21, 22, 39, 40, 47, 48, 50, 51, 53, 54, 65, 66, 78 and 79 and amended claims 1, 8, 20, 30, 35, 36, 37, 42, 43, 49, 52, 64 and 74. The amendments were accepted by the Examiner.

On July 15, 2003, Appellant cancelled claims 32 and 33 and amended claims 1, 30 and 31. The amendments were accepted by the Examiner.

Claims 1-5, 8, 11-20, 23-31, 34-38, 41-46, 49, 52, 55-64 and 67-77 received a final rejection on September 18, 2003.

No further amendments were made.

A Notice of Appeal was filed on January 20, 2004.

Summary of the Invention

Appellant describes a mechanism for maintaining cache coherence by selectively resetting a portion of memory. Appellant teaches, at p. 7, line 24- p. 8, line 21, and claims in claims 1-5, 8, 11-20, 23-31, 34-38, 41-46, 49, 52, 55-64 and 67-77, that if one can determine if the processor accessing a resource is the last processor to access the resource, it is not necessary to reset cache in that processor associated with that resource. Instead, the processor can proceed using the cached data.

Appellant further teaches, and claims in claims 1-5, 64 and 67-73, that the above technique can be used advantageously across a plurality of processors as a more efficient way to maintain cache coherence.

Appellant also teaches on p. 7, and claims in claims 8, 11-20, 23-29, 42-46, 49, 52, 55-63 and 74-77, that a switch mechanism can be used to coordinate the selective resetting on memory.

Appellant also describes in Fig. 5 and at p. 12, line 11 through p. 14, line 6 and claims in claims 30, 31, and 34-36, a data structure that can be used to implement such a cache coherence mechanism.

Finally, Appellant describes in Fig. 6 and at p. 14, line 7- p. 15, line 4, and claims in claims 37, 38 and 41, a method of sharing a resource according to the present invention.

Issues Presented for Review

1. Are claims 1-5, 8, 11, 14-20, 26-31, 34, 36-38, 41-46, 49, 52, 55, 58-59, 60-64 and 70-77 properly rejected under 35 U.S.C.§103(a) as being unpatentable over Lange et al. 9US 3,845,474) in view of Averill (US 5,313,591)?

2. Are claims 12, 23-25, 56, 67-79 properly rejected under 35 U.S.C.§103(a) as being unpatentable over Lange in view of Averill, and in further view of the Structured Computer Organization by Andrew Tanenbaum?

3. Are claims 13 and 57 properly rejected under 35 U.S.C.§103(a) as being unpatentable over Lange in view of Averill, and in further view of Tanenbaum and Georgiou et al. (US 4,633,394)?

4. Is claim 35 properly rejected under 35 U.S.C.§103(a) as being unpatentable over the combination of the Lange, Averill, and Gusefski (US 5,202,972)?

Grouping of Claims

The claims are grouped and argued as follows. Claims 1-5, 64 and 67-73 stand or fall together. Claims 8, 11-20, 23-29, 42-46, 49, 52, 55-63 and 74-77 stand or fall together. Claims 30, 31, and 34-36 stand or fall together. Claims 37, 38 and 41 stand or fall together.

Argument

§103 Rejection of the Claims

Claims 1-5, 8, 11, 14-20, 26-31, 34, 36-38, 41-46, 49, 52, 55, 58-59, 60-64 and 70-77 were rejected under 35 USC § 103(a) as being unpatentable over Lange et al. (US 3,845,474) in view of Averill (US 5,313,591).

1) The Applicable Law

The Examiner has the burden under 35 U.S.C. § 103 to establish a *prima facie* case of obviousness. *In re Fine*, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). To do that the Examiner must show that some objective teaching in the prior art or some knowledge generally available to one of ordinary skill in the art would lead an individual to combine the relevant teaching of the references. *Id.*

The *Fine* court stated that:

> Obviousness is tested by "what the combined teaching of the references would have suggested to those of ordinary skill in the art." *In re Keller*, 642 F.2d 413, 425, 208 USPQ 871, 878 (CCPA 1981)). But it "cannot be established by

> combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination." *ACS Hosp. Sys.*, 732 F.2d at 1577, 221 USPQ at 933. And "teachings of references can be combined *only* if there is some suggestion or incentive to do so." *Id.* (emphasis in original).

The M.P.E.P. adopts this line of reasoning, stating that

> In order for the Examiner to establish a *prima facie* case of obviousness, three base criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure.

M.P.E.P. § 2142 (citing *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir. 1991)). In addition, the teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991); MPEP § 2143. The Examiner must avoid hindsight. *In re Bond*, 910 F.2d 831, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990).

2) Applying the Law

As the Examiner notes in the Office Action, Lange discusses the clearing of cache store each time that a processor arbitrates for access to main memory store. As noted in the response to the response to the first Office Action, this is a fairly simple mechanism. It has the disadvantage that, if a processor wins consecutive arbitrations for main memory, its cache is still invalidated. Appellant avoids that problem by monitoring the present and previous owners of memory or sections of memory in main memory, and clearing cache in the processor accessing that memory only if it is not the last processor to access that memory or section of memory.

The Examiner stated that "Lange does not explicitly state resetting the current owner's memory when the current owner is a different entity from the previous owner." This statement is not completely accurate. As can be seen at col. 11, line 50 through col. 12, line 18, Lange describes a mechanism that can be used by a first processor to lock a second processor from a portion of memory and to clear any data cached by the second processor from that portion of memory. As noted at col. 12, lines 4-5, whenever the first processor locks the memory, it uses the READ CLEAR signal to control the clearing of the second processor's cache store.

It should be noted that the clearing of the second processor's cache store is done by the first processor. There is, therefore, no "means for identifying a previous owner of the resource" as taught by Appellant and claimed in claims 1-5. Since there is no "means for identifying a previous owner of the resource," Lange cannot have "a memory, wherein at least a portion of the memory of the at least one entity is selectively reset when the at least one entity has access to the resource, wherein the memory portion is reset if the entity is not the previous owner of the resource, and wherein the memory portion is not reset when the at least one entity is the previous owner of the resource" as taught by Appellant and claimed in claims 1-5.

Averill addresses the problem of designing computer busses to accommodate transactions which may take a variable amount of time to complete. Averill states that a cache coherency mechanism which relies on the flushing or purging of cache memory of other processors as part of a transaction originated by one of the processors is an example of such a transaction. As such, Averill adds little to Lange.

In the Final Office Action, the Examiner stated

> In response to Applicant's argument that Averill does not teach a system which discriminates between consecutive and disparate data owners in determining whether to reset a portion of memory associated with the processor (cite omitted): Averill does teach this. Averill discloses that it is know to keep each processor's cache accurate and current (column 1, lines 25-26), and purge or flush must be done before any new processor gains control of the bus (column 1, lines23-42). Averill's system recognizes the processor with gaining new control

> and resetting its cache accordingly, which is discriminating between consecutive and disparate data owners in determining whether to reset a portion of memory.

Final Office Action, p. 17, lines 3-11. The section the Examiner cites is reproduced below:

> Processors in a multiprocessor system may have local cache memory for fast access to information. The information in each processor's cache must be accurate and current. This is called cache coherency. A shared-memory cache-coherent multiprocessor system has multiple processors, each with local cache memory and a system for maintaining coherency of each local cache. In a shared-memory cache-coherent multiprocessor system, if any processor changes data in its cache, subsequent accesses of the same data by other processors must reflect the change. Therefore, the change must propagate to all other caches in the system. When a processor gains control of the bus, other processors may not need to do anything, or other processors may have to flush or purge and later refill cache memory. The flush or purge must be accomplished for each cache in all processors in the system before the original transaction can be considered complete. Therefore, in general, transactions which affect cache memory require a variable amount of time.

Averill, col. 1, lines 23-42. Appellant respectfully submits that the section that the Examiner has highlighted simply states the coherency problem; it offers no solution. The Examiner has failed to point out the "means for identifying a previous owner of the resource"; there is no requirement in Averill for such a means, since Averill does not disclose determining if a processor was the previous owner of the resource, and deciding whether to purge or not based on that determination.

Neither Lange nor Averill teach a system which discriminates between consecutive and disparate data owners in determining whether to reset a portion of memory associated with the processor, as is taught by Appellant and claimed in claims 1-5 and 42. At the same time, neither Lange nor Averill disclose a system which identifies the present and past owner of a shared resource, as is taught by Appellant and claimed in claims 30, 31, 34-38 and 41. Appellant respectfully submits that claims 1-5, 30, 31, 34-38, 41 and 42 do distinguish over Lange et al.

and Averill, either alone or together. Reversal of the Examiner's rejection of claims 1-5, 30, 31, 34-38, 41 and 42 is respectfully requested.

Similarly, neither Lange nor Averill teach a system which discriminates between consecutive ownership of a resource by the same processor versus consecutive ownership of a resource by the different processors in determining whether to reset a portion of memory associated with one of the processors, as is taught by Appellant and claimed in claims 8, 11-20, 23, 42-46, 49, 52, 55-64, and 67-77. Reversal of the Examiner's rejection of claims 8, 11-20, 23, 42-46, 49, 52, 55-64, and 67-77 is respectfully requested.

Claims 12, 23-25, 56 and 67-69 were rejected under 35 USC § 103(a) as being unpatentable over Lange et al. (US 3,845,474) in view of Averill (US 5,313,591), and further in view of Tanenbaum, Structured Computer Organization. Claims 12, 23-25, 56 and 67-69 are dependent claims. The claims on which they depend are patentable for the reasons given above.

In addition, the decision as to how to implement, for instance, the lock or the software switch, is a decision driven by factors disclosed as part of the description given of Appellant's invention. The Examiner is taking Official Notice that these implementation details would be obvious to one trying to implement the invention described by Appellant without the signposts provided by Appellant in his disclosure. The Examiner is, therefore, relying on impermissible hindsight, or is relying on an uncited reference or personal knowledge. Either way, reversal of the Examiner's rejection of claims 12, 23-25, 56 and 67-69 is respectfully requested.

Claims 13 and 57 were rejected under 35 USC § 103(a) as being unpatentable over Lange et al. (US 3,845,474) in view of Averill (US 5,313,591), and further in view of Tanenbaum and Georgiou et al. (US 4,633,394). Claims 13 and 57 are dependent claims. The claims on which they depend are patentable for the reasons given above.

In addition, as noted above, the decision as to how to implement the software switch is a decision driven by factors disclosed as part of the description given of Appellant's invention. The Examiner is taking Official Notice that these implementation details would be obvious to

one trying to implement the invention described by Appellant without the signposts provided by Appellant in his disclosure. The Examiner is, therefore, either relying on impermissible hindsight, or relying on an uncited reference or personal knowledge to supply the suggestion or teaching to combine Georgiou with the other references to form a disclosure which teaches a switch mechanism which is based on a Djikstra primitive. Reversal of the Examiner's rejection of claims 13 and 57 is respectfully requested.

Claim 35 was rejected under 35 USC § 103(a) as being unpatentable over Lange et al. (US 3,845,474), Averill (US 5,313,591) and Gusefski (US 5,202,972).

Lange and Averill are discussed above.

As noted in the response to the previous Office Action, Gusefski describes a system having processors and common master storage unit. Each processor includes cache memory. The common master storage unit includes master storage (102, 106), a second level of cache (27) and a storage controller (26) that arbitrates for access to master storage. Col. 4, lines 41-66. If a processor tries to retrieve data that it is not within its L1 cache, it accesses storage controller 26. If the master storage resource is being used by another processor, it waits for its turn. Eventually, it does access storage controller 26 and can retrieve data. The system described by Gusefski requires that each processor monitor for fetch conflicts in order to maintain cache coherency. Fig. 6 and col. 14, lines 23-58. As noted at col. 14, lines 10-14, this monitoring is done by comparing the absolute address of the storage operand against the absolute address portion of the store buffer AA 20-32b entries whose EOI bits are one and C bits are zero.

Claim 35 is dependent on claim 30 and patentable for the reasons given for claim 30 above.

In addition, the Examiner has failed to establish a *prima facie* case of obviousness by failing to show a data type in any of the cited references which is "adapted to represent at least one portion of the resource, wherein the data type includes at least one location of the at least one

portion of the resource and at least one dimension of the at least one portion of the resource" as taught by Appellant and claimed in claim 35.

Conclusion

It is respectfully submitted that the claimed invention is not unpatentable in view of the cited art. It is respectfully submitted that claims 1-5, 8, 11-20, 23-31, 34-38, 41-46, 49, 52, 55-64 and 67-77 should therefore be allowed. Reversal of the Examiner's rejections of claims 1-5, 8, 11-20, 23-31, 34-38, 41-46, 49, 52, 55-64 and 67-77 is respectfully requested.

Respectfully submitted,

MARK D. RUSTAD

By his Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. Box 2938
Minneapolis, MN 55402
(612) 373-6909

Date May 20, 2004 By Thomas F. Brennan

Thomas F. Brennan
Reg. No. 35,075

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Mail Stop APPEAL BRIEF - PATENTS, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this 20th day of May, 2004.

CANDIS BUENDING
Name

Candis Buending
Signature

APPENDIX: The Claims on Appeal

Claims 1-5, 8, 11-20, 23-31, 34-38, 41-46, 52, 55-64 and 67-77 are on appeal.

1. A system comprising:

a bus;

a resource coupled to the bus;

means for identifying a previous owner of the resource; and

a plurality of entities coupled to the bus, at least one entity among the plurality of entities including a memory, wherein at least a portion of the memory of the at least one entity is selectively reset when the at least one entity has access to the resource, wherein the memory portion is reset if the entity is not the previous owner of the resource, and wherein the memory portion is not reset when the at least one entity is the previous owner of the resource.

2. The system of claim 1, wherein the at least one entity is an integrated circuit.

3. The system of claim 1, wherein the resource includes at least a portion of a memory device.

4. The system of claim 1, further comprising a manager to manage at least one request from the plurality of entities to access the resource.

5. The system of claim 1, further comprising an arbiter coupled to the plurality of entities to arbitrate at least one bus request from the plurality of entities.

6. (Cancelled)

7. (Cancelled)

8. An integrated circuit for allowing a resource to be controlled by a plurality of processors including a first and second processor, wherein the first processor includes a fast memory, the integrated circuit comprising:

a bus;

a central computing unit coupled to the bus; and

a switch mechanism, coupled to the central computing unit, to switch the control of the resource, wherein a portion of the fast memory of the first processor is reset when the control of the resource is switched to the second processor and not reset when control of the resource remains with the first processor.

9. (Cancelled)

10. (Cancelled)

11. The integrated circuit of claim 8, wherein the switch mechanism is a hardware device.

12. The integrated circuit of claim 8, wherein the switch mechanism is a software switch.

13. The integrated circuit of claim 12, wherein the software switch is a Djikstra primitive.

14. The integrated circuit of claim 8, wherein the at least one resource is a hardware resource.

15. The integrated circuit of claim 14, wherein the hardware resource is a memory.

16. The integrated circuit of claim 8, further comprising a communications channel controller coupled to the bus.

17. The integrated circuit of claim 8, wherein the at least one resource is a software resource.

18. The integrated circuit of claim 17, wherein the software resource is a data structure.

19. The integrated circuit of claim 8, wherein the fast memory is cache memory.

20. An integrated circuit for allowing a resource to be shared by a plurality of processors, including a first and second processor, wherein the first processor includes a fast memory, the integrated circuit comprising:

a bus;

a central computing unit coupled to the bus; and

a lock coupled to the central computing unit to reserve exclusive control of the resource, wherein a portion of the fast memory of the first processor is reset when the second processor obtains exclusive control of the resource from the first processor and not reset when exclusive control remains with the first processor.

21. (Cancelled)

22. (Cancelled)

23. The integrated circuit of claim 20, wherein the lock is a hardware register.

24. The integrated circuit of claim 20, wherein the lock is a software semaphore.

25. The integrated circuit of claim 24, wherein the software semaphore is a binary semaphore.

26. The integrated circuit of claim 20, further comprising a communications channel controller coupled to the bus.

27. The integrated circuit of claim 20, wherein the fast memory is cache memory.

28. The integrated circuit of claim 27, wherein the cache memory is primary cache memory.

29. The integrated circuit of claim 27, wherein the cache memory is secondary cache memory.

30. A data structure in a machine-readable medium for allowing a resource to be shared among a plurality of processors, at least one processor of the plurality of processors including a fast memory, the data structure comprising:

a state for indicating that the resource is under control;

a first identifier for identifying a past processor that had exclusive control of the resource; and

a second identifier for identifying a present processor that has exclusive control of the resource.

31. The data structure of claim 30, wherein the data structure is a class, the data structure further comprising means for comparing the first identifier to the second identifier and means for resetting at least a portion of the fast memory of the present processor when the first identifier does not match the second identifier.

32. (Cancelled)

33. (Cancelled)

34. The data structure of claim 30, wherein the fast memory is cache memory.

35. The data structure of claim 30, further comprising a data type that is adapted to represent at least one portion of the resource, wherein the data type includes at least one location of the at least one portion of the resource and at least one dimension of the at least one portion of the resource.

36. The data structure of claim 30, further comprising a list that includes at least one location of at least one portion of the resource and at least one dimension of at least one portion of the resource.

37. A method for allowing at least one resource to be shared among a plurality of processors, the method comprising:

obtaining exclusive control over the at least one resource by a present processor, the present processor including a fast memory;

identifying a past processor to obtain a first identity, wherein the past processor had exclusive control over the at least one resource;

identifying a present processor to obtain a second identity, the present processor having exclusive control over the at least one resource;

comparing the first identity and the second identity so as to determine if the present processor is different from the past processor; and

resetting a portion of the fast memory of the present processor when the past processor is different from the present processor and not resetting a portion of the fast memory of the present processor when the past processor is the same as the present processor.

38. The method of claim 37, wherein identifying the present processor further comprises the fast memory as cache memory.

39. (Cancelled)

40. (Cancelled)

41. The method of claim 37, wherein the progression of the method is in the order presented.

42. A method for scheduling access to a resource from among a plurality of processors, the method comprising:

obtaining access to the resource by a requesting processor, the requesting processor including a cache memory;

excluding access to the resource from the plurality of processors except for the requesting processor; and

resetting a portion of the cache memory of the requesting processor when the requesting processor is different from a processor that previously had access to the resource, and not

resetting a portion of the cache memory of the requesting processor when the requesting processor is the same as a processor that previously had access to the resource.

43. An integrated circuit for allowing a resource to be controlled by a plurality of processors, including a first and second processor, wherein the first processor includes a fast memory, the integrated circuit comprising:

a bus;

a central computing unit coupled to the bus;

a switch mechanism for switching the control of the resource; and

a lock, in a cooperative relationship with the switching mechanism, for reserving exclusive control of the resource to the first processor, wherein at least a portion of the fast memory of the first processor is reset when the second processor obtains exclusive control of the resource and not reset when the exclusive control of the resource remains with the first processor.

44. The integrated circuit of claim 43, wherein the fast memory is cache memory.

45. The integrated circuit of claim 43, further comprising a communications channel controller coupled to the bus, wherein the communications channel controller is receptive to diverse communications protocols.

46. The integrated circuit of claim 43, wherein the cooperative relationship of the switch mechanism and the lock maintains cache coherency.

47. (Cancelled)

48. (Cancelled)

49. An integrated circuit for allowing a resource to be controlled by a plurality of processors, including a first and second processor, wherein the first processor includes a fast memory, the integrated circuit comprising:

a bus;

a central computing unit coupled to the bus; and

a scheduler, coupled to the central computing unit, for scheduling the control of the resource, wherein a portion of the fast memory of the first processor is reset when the resource becomes under the control of the second processor and not reset when the resource remains under the control of the first processor.

50. (Cancelled)

51. (Cancelled)

52. A system comprising:

a bus;

a resource coupled to the bus;

a plurality of processors coupled to the bus including a first and second processor, wherein the first processor includes a fast memory; and

a switch mechanism, coupled to the bus, to switch the control of the resource, wherein a portion of the fast memory of the first processor is reset when the control of the resource is switched to the second processor and not reset when control of the resource remains with the first processor.

53. (Cancelled)

54. (Cancelled)

55. The system of claim 52, wherein the switch mechanism is a hardware device.

56. The system of claim 52, wherein the switch mechanism is a software switch.

57. The system of claim 56, wherein the software switch is a Djikstra primitive.

58. The system of claim 52, wherein the at least one resource is a hardware resource.

59. The system of claim 58, wherein the hardware resource is a memory.

60. The system of claim 52, wherein the at least one processor includes a communications channel controller.

61. The system of claim 52, wherein the at least one resource is a software resource.

62. The system of claim 61, wherein the software resource is a data structure.

63. The system of claim 52, wherein the fast memory is cache memory.

64. A system comprising:

a bus;

a resource coupled to the bus;

a plurality of processors coupled to the bus including a first and second processor, wherein the first processor includes a fast memory; and

a lock to reserve exclusive control of the resource, wherein a portion of the fast memory of the first processor is reset when the second processor of the plurality of processors obtains exclusive control of the resource from the first processor and not reset when exclusive control of the resource remains with the first processor.

65. (Cancelled)

66. (Cancelled)

67. The system of claim 64, wherein the lock is a hardware register.

68. The system of claim 64, wherein the lock is a software semaphore.

69. The system of claim 68, wherein the software semaphore is a binary semaphore.

70. The system of claim 64, further comprising a communications channel controller coupled to the bus.

71. The system of claim 64, wherein the fast memory is cache memory.

72. The system of claim 71, wherein the cache memory is primary cache memory.

73. The system of claim 71, wherein the cache memory is secondary cache memory.

74. A system comprising:

a bus;

a resource coupled to the bus;

a plurality of processors coupled to the bus including a first and second processor, wherein the first processor includes a fast memory;

a switch mechanism to switch the control of the resource; and

a lock, in a cooperative relationship with the switching mechanism, to reserve exclusive control of the resource to a first processor, wherein at least a portion of the fast memory of the first processor is reset when the second processor obtains exclusive control of the resource and not reset when the exclusive control of the resource remains with the first processor.

75. The system of claim 74, wherein the fast memory is cache memory.

76. The system of claim 74, wherein the at least one processor includes a communications channel controller, wherein the communications channel controller is receptive to diverse communications protocols.

77. The system of claim 74, wherein the cooperative relationship of the switch mechanism and the lock maintains cache coherency.

78. (Cancelled)

79. (Cancelled)